Exercícios

Exercícios

SOLID - LSP

Exercício 1: Formas geométricas

Refatore o código a seguir utilizando os princípios :

- **Substituição de Liskov (LSP)**

```
class Retangulo {
    protected int largura;
    protected int altura;

    public void setLargura(int largura) {
        this.largura = largura;
    }

    public void setAltura(int altura) {
        this.altura = altura;
    }

    public int getArea() {
        return largura * altura;
    }
}

class Quadrado extends Retangulo {
    @Override
    public void setLargura(int largura) {
        this.largura = largura;
        this.altura = largura;    }

    @Override
    public void setAltura(int altura) {
        this.altura = altura;
        this.largura = altura;    }
}
```

```
public class Main {
    public static void main(String[] args) {
        Retangulo r = new Quadrado();
        r.setLargura(5);
        r.setAltura(10);

        System.out.println("Área esperada: 5 * 10 = 50,
mas retorna: " + r.getArea());
    }
}
```

Exercícios

SOLID - LSP

Exercício 2: Contas Bancárias

Refatore o código a seguir utilizando os princípios :

- **Substituição de Liskov (LSP)**

```
interface ContaBancaria {
    void sacar(double valor);
    double getSaldo();
}

class ContaCorrente implements ContaBancaria {
    protected double saldo;

    public ContaCorrente(double saldoInicial) {
        this.saldo = saldoInicial;
    }

    public void sacar(double valor) {
        saldo -= valor;
}

    public double getSaldo() {
        return saldo;
    }
}
```

```
class ContaPoupanca implements ContaBancaria {
    private double saldo;

    public ContaPoupanca(double saldoInicial) {
        this.saldo = saldoInicial;
    }

    public void sacar(double valor) {
        if (valor > saldo) {
            System.out.println("Saldo insuficiente para saque!");
            return;
        }
        saldo -= valor;
    }

    public double getSaldo() {
        return saldo;
    }
}

public class Main {
    public static void main(String[] args) {
        ContaBancaria conta1 = new ContaCorrente(100);
        ContaBancaria conta2 = new ContaPoupanca(100);

        conta1.sacar(150);
        conta2.sacar(150);
        System.out.println("Saldo da Conta Corrente: "+ conta1.getSaldo());
        System.out.println("Saldo da Conta Poupança: "+ conta2.getSaldo());
    }
}
```

Exercício 3: Gestão de Estoque de Produtos com Categorias Variadas

Uma loja possui um estoque de produtos que inclui eletrônicos, roupas e alimentos. Cada categoria tem suas próprias regras:

- **Eletrônicos:** Devem ter uma garantia de fábrica e podem ser devolvidos em até 30 dias.
- **Roupas:** Possuem tamanhos e cores diferentes e só podem ser devolvidas se não forem usadas.
- **Alimentos:** Possuem uma data de validade e não podem ser devolvidos após a compra.

O sistema deve ser capaz de armazenar diferentes tipos de produtos, garantindo que novos tipos possam ser adicionados no futuro sem modificar a estrutura do código existente.

Desafio:
Crie um gerenciador de estoque que possa lidar com múltiplas categorias de produtos, respeitando suas regras específicas e permitindo a adição de novas categorias sem impactar as já existentes.

Exercício 4: Cálculo de Frete Inteligente para Diferentes Modalidades de Entrega

Um e-commerce precisa calcular o valor do frete baseado no método de entrega escolhido pelo cliente. As opções disponíveis são:

- **Correios:** Cálculo baseado no peso e na distância do destino.
- **Transportadora:** Aceita apenas pedidos acima de um certo valor e possui taxas fixas por região.
- **Retirada na Loja:** Não possui custo de frete, mas exige que o estoque da unidade selecionada seja verificado.

Novas formas de entrega podem ser adicionadas no futuro, como entrega via drone ou coleta em lockers.

Desafio:
Implemente um sistema de cálculo de frete que suporte múltiplas modalidades de entrega e permita a inclusão de novas opções de frete sem modificar a lógica central do sistema.

Exercício 5: Processamento de Pedidos com Status Dinâmicos

Uma loja precisa de um gerenciador de pedidos que acompanhe o fluxo de um pedido desde a compra até a entrega. Um pedido pode passar pelos seguintes estados:

- **Pendente:** Pedido aguardando pagamento.
- **Pago:** Pagamento confirmado, aguardando separação do estoque.
- **Em Transporte:** Pedido enviado para entrega.
- **Entregue:** Pedido recebido pelo cliente.

No futuro, novos estados podem ser adicionados (exemplo: Pedido Cancelado, Pedido Devolvido).

Desafio:
Desenvolva um sistema de processamento de pedidos que possa lidar com diferentes estados e seja facilmente estendido para suportar novas situações sem impactar a lógica principal.